```
> d his full

     (FILE 'USPAT' ENTERED AT 15:13:05 ON 25 FEB 1999)
L1   (       201)SEA 707/203/CCLS
L2           201 SEA 707/203/CCLS
L3           201 SEA 707/203/CCLS
L4             0 SEA L3 AND LOGICAL(W)PRIMARY(W)KEY
L5             7 SEA L3 AND PRIMARY(W)KEY
L6             0 SEA L3 AND EFFECTIVE(W)VALUE
L7             0 SEA L3 AND TARGET(W)EFFECTIVE


     FILE USPAT
       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
       *                  W E L C O M E   T O   T H E                *
       *           U. S.   P A T E N T   T E X T   F I L E           *
       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
*          6:30am - [redacted]0pm Monday through Friday          *
*          7:30am - [redacted]0pm Saturday, Sunday, Holidays[redacted]          *
*                                                                *
*      APS is unavailable Thanksgiving Day, Christmas Day,       *
*      and New Year's Day.                                       *
*                                                                *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
FILE 'USPAT' ENTERED AT 15:13:05 ON 25 FEB 1999

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*              W E L C O M E   T O   T H E                      *
*         U. S.   P A T E N T   T E X T   F I L E               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

=> s 707/203/ccls

L1          201 707/203/CCLS
COMMAND INTERRUPTED
'L1 ' HAS NO ANSWERS

=> s 707/203/ccls

L2          201 707/203/CCLS

=> s 707/203/ccls

L3          201 707/203/CCLS

=> s l3 and logical(w)primary(w)key

          64893 LOGICAL
         457142 PRIMARY
         165907 KEY
              0 LOGICAL(W)PRIMARY(W)KEY
L4            0 L3 AND LOGICAL(W)PRIMARY(W)KEY

=> s l3 and primary(w)key

         457142 PRIMARY
         165907 KEY
            413 PRIMARY(W)KEY
L5            7 L3 AND PRIMARY(W)KEY

=> d l5 -7


1.  5,873,096, Feb. 16, 1999, Method of maintaining a network of
partially replicated database system; Peter S. Lim, et al., 707/201, 3,
5, 102, 203 [IMAGE AVAILABLE]

2.  5,802,514, Sep. 1, 1998, Automated client/server development tool
using drag-and-drop metaphor; Val J. Huber, 707/4; 345/348; 395/701;
707/1, 2, 10, 101, 102, 203, 505, 506, 507, 508 [IMAGE AVAILABLE]

3.  5,706,499, Jan. 6, 1998, Functional compensation in a heterogeneous,
distributed database environment; James C. Kleewein, et al., 707/10;
340/825.34; 395/182.1, 187.01, 200.59; 707/4, 201, 203 [IMAGE
AVAILABLE]

4.  5,603,024, Feb. 11, 1997, Lossless distribution of time series data
in a relational data base network; Robert D. Goldring, 707/203;
364/222.81, 282.1, DIG.1 [IMAGE AVAILABLE]

5.  5,440,730, Aug. 8, 1995, Time index access structure for temporal
databases having concurrent multiple versions; Ramez A. Elmasri, et al.,
```

707/203; 364/282.1, 282.[redacted] DIG.1; 711/159 [IMAGE AVAILABLE]

6. 5,280,612, Jan. 18, 1994, Multiple version database concurrency control system; Raymond A. Lorie, et al., 707/8; 364/222.81, 222.82, 246.6, 246.8, 259, 259.2, 261, 262.4, 262.9, 264, 281.1, 281.3, 281.4, 281.5, 282.1, 282.4, DIG.1; 395/730; 707/9, **203**, 206 [IMAGE AVAILABLE]

7. 4,627,019, Dec. 2, 1986, Database management system for controlling concurrent access to a database; Fred K. Ng, 707/8; 364/920, 929.1, 942.3, 942.4, 942.5, 948.3, 948.31, 957, 957.1, 960, 960.5, 962, 962.1, 974, 974.1, 974.4, 975, 976, 976.1, DIG.2; **707/203** [IMAGE AVAILABLE]

> s 707/203/ccls

L12 201 707/203/CCLS

=> s l12 and primary(w)key

457142 PRIMARY
165907 KEY
413 PRIMARY(W)KEY
L13 7 L12 AND PRIMARY(W)KEY

=> d l13 -7 ab

US PAT NO: 5,873,096 [IMAGE AVAILABLE] L13: 1 of 7

ABSTRACT:
A partially replicated database is maintained so that updates made to a central database, or to another partially replicated database, are selectively propagated to the partially replicated database. Updates are propagated to a partially replicated database if the owner of the partially replicated database has "visibility" to the data being updated. Visibility is determined by using predetermined rules stored in a rules database. Typically, the stored rules are assessed against data content of a plurality of tables making up a single logical entity, known as a docking object, that is being updated.

US PAT NO: 5,802,514 [IMAGE AVAILABLE] L13: 2 of 7

ABSTRACT:
A tool for the development of multiple-table database applications for client/server environments automates both capture of system requirements and code production. A client portion of a multiple-table, client/server database application for processing requests against a server database, is developed by first storing in a repository a description of the server database describing database entities within the server database and relationships between those database entities. Representations of the database entities are displayed, and an application drawing window is provided. The user drags and drops within the application drawing window one of said representations that represents a particular database entity. The tool then creates within the repository an entry for the particular database entity, and draws within the drawing window a representation of the particular database entity. For each database entity for which an entry within the repository has been created, the tool checks the description of the server database stored in the repository to determine whether a relationship exists between the particular database entity and the database entity being checked. If a relationship does exist between the particular database entity and the database entity being checked, the tool then creates within the repository an entry for that relationship, and draws within the drawing window a connector representing that relationship. The foregoing drag-and-drop sequence is repeated multiple times. When the design is complete, the tool, using information stored in the repository, automatically generates the client portion of the multiple-table, client/server database application.

US PAT NO: 5,706,499 [IMAGE AVAILABLE] L13: 3 of 7

ABSTRACT:
A system and method for compensating for functional differences between

heterogeneous database management systems, wherein data associated with a client is distributed among the heterogeneous database management systems, is discussed. The system simulates support of multiple pending actions on a single connection in any of the heterogeneous database management systems which does not support multiple pending actions on a single connection. Also, the system: (1) simulates support of cursors declared "with hold" in any of the heterogeneous database management systems which does not support cursors declared "with hold"; (2) simulates support of positioned update actions in any of the heterogeneous database. management systems which does not support positioned update actions; (3) simulates support of host variables in any of the heterogeneous database management systems which does not support host variables; and (4) compensates for security log-in procedure differences between the heterogeneous database management systems.

US PAT NO: 5,603,024 [IMAGE AVAILABLE] L13: 4 of 7

ABSTRACT:
A computer processing system that receives sequences of changes to a data base and records them into an activity log for later retrieval also maintains a consistent change data table that contains sufficient change information for each of the changes to the data base such that the changes can be propagated through multiple copies of the data base by consulting the consistent change data table. The consistent change data includes information sufficient to permit reconstruction of the data base to reflect the condition of the data base at any moment of time in the activity log. Because the consistent change data is complete, it permits producing multi-generational copies of data base tables for replication from one copy level to any other subsequent level, or iteration, of copy.

US PAT NO: 5,440,730 [IMAGE AVAILABLE] L13: 5 of 7

ABSTRACT:
A time index for temporal databases is provided which enables the retrieval of database object versions that are valid during specified time periods. Unlike prior access and retrieval structures, the present index is based on objects whose search values are time intervals rather than time points. A series of ordered indexing points is defined by the start and end of object version intervals and these points are used to build an indexing structure, which may take the form of a B.sup.+ -tree. Each leaf node entry of the B.sup.+ -tree represents an indexing point and has an associated bucket of pointers which identify all object versions that are valid at that time. Storage space is reduced by including only incremental change indicators in the buckets of non-leading leaf entries and calculating needed pointers from such indicators. The time index may be employed in multi-level structures with attribute indexes to greatly improve the efficiency of temporal search operations, such as aggregate functions and temporal selection, as well WHEN and JOIN operators.

US PAT NO: 5,280,612 [IMAGE AVAILABLE] L13: 6 of 7

ABSTRACT:
An improved concurrency control system for application to a distributed concurrent transaction and query processing system using multi-version database records to overcome delays arising from lock conflicts. Read-only queries are afforded a consistent "stable state" of the database during the life of the query. Updating transactions requiring locks can proceed without waiting for the termination of long queries. At least two database versions are necessary, although availability of more versions permits long read-only queries to phase-out over time without forcing new queries to use aged "stable-state" data and without roll-back. Read-only queries can be terminated and converted to locking transactions to permit an update of the "stable state" database version before the queries would normally terminate. A novel record key structure

having a plurality of su[redacted]ructures corresponding to the [redacted]veral database versions is used to acce[redacted] database records. Rapid select[redacted] of proper record version and efficient version tracking and updating is effected using several bit-mapped transaction index tables.

US PAT NO: 4,627,019 [IMAGE AVAILABLE] L13: 7 of 7

ABSTRACT:
A method of assuring that each of a plurality of contemporaneously active database transactions comprising at least one read transaction and at most one update transaction has a consistent view of a database storing a plurality of versions of a relation. A transaction has a consistent view of a database if the data available to a transaction are not changed during its execution. An access dictionary is stored comprising an array of access blocks each defining the database location of one of the relation versions. At any given time, only one of the relation versions is defined as current. A relation dictionary comprising an array of relation blocks is stored such that as each database transaction is begun, a relation block associated with that database transaction is stored defining the access block defining the database location of the relation version then defined as current. For the update transaction, a new access block in the access dictionary is stored defining a new database location to be used for storing a new relation version. The relation block associated with the update transaction is modified to define the new access block and the new relation version is stored in the new database location. In addition the current relation version is redefined as old and the new relation version is defined as current. Access to the database by each of the plurality of database transactions is permitted only via the relation block associated with that database transaction. The method can be extended to allow contemporaneous access by noninterfering writers and an arbitrary number of readers to a database storing a plurality of relations each having a plurality of versions.